

CURSO DE JAVA CON JDBC

EJERCICIO

MANEJO DE TRANSACCIONES EN JDBC



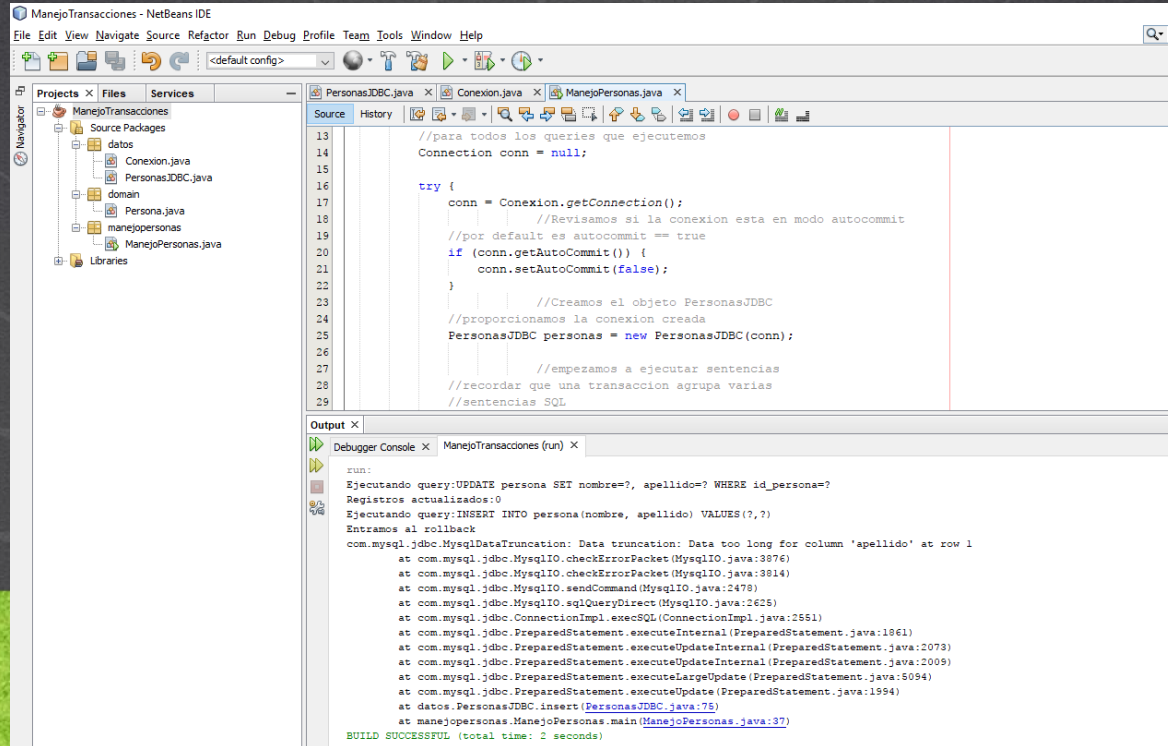
Experiencia y Conocimiento para tu vida

CURSO DE JAVA CON JDBC

www.globalmentoring.com.mx

OBJETIVO DEL EJERCICIO

Crear un programa para practicar el concepto de transacciones con JDBC. Al finalizar deberemos observar lo siguiente:



```
13 //para todos los queries que ejecutemos
14 Connection conn = null;
15
16 try {
17     conn = Conexion.getConnection();
18     //Revisamos si la conexion esta en modo autocommit
19     //por default es autocommit == true
20     if (conn.getAutoCommit() == true) {
21         conn.setAutoCommit(false);
22     }
23     //Creamos el objeto PersonasJDBC
24     //proporcionamos la conexion creada
25     PersonasJDBC personas = new PersonasJDBC(conn);
26
27     //empezamos a ejecutar sentencias
28     //recordar que una transaccion agrupa varias
29     //sentencias SQL
```

Output

Debugger Console x ManejoTransacciones (run) x

run:

Ejecutando query:UPDATE persona SET nombre=?, apellido=? WHERE id_persona=?

Registros actualizados:0

Ejecutando query:INSERT INTO persona(nombre, apellido) VALUES(?,?)

Entramos al rollback

com.mysql.jdbc.MySQLDataTruncation: Data truncation: Data too long for column 'apellido' at row 1

at com.mysql.jdbc.MySQLIO.checkErrorPacket(MySQLIO.java:3876)

at com.mysql.jdbc.MySQLIO.checkErrorPacket(MySQLIO.java:3814)

at com.mysql.jdbc.MySQLIO.sendCommand(MySQLIO.java:2478)

at com.mysql.jdbc.MySQLIO.sqlQueryDirect(MySQLIO.java:2625)

at com.mysql.jdbc.ConnectionImpl.execSQL(ConnectionImpl.java:2551)

at com.mysql.jdbc.PreparedStatement.executeInternal(PreparedStatement.java:1861)

at com.mysql.jdbc.PreparedStatement.executeUpdateInternal(PreparedStatement.java:2073)

at com.mysql.jdbc.PreparedStatement.executeUpdateInternal(PreparedStatement.java:2009)

at com.mysql.jdbc.PreparedStatement.executeLargeUpdate(PreparedStatement.java:5094)

at com.mysql.jdbc.PreparedStatement.executeUpdate(PreparedStatement.java:1994)

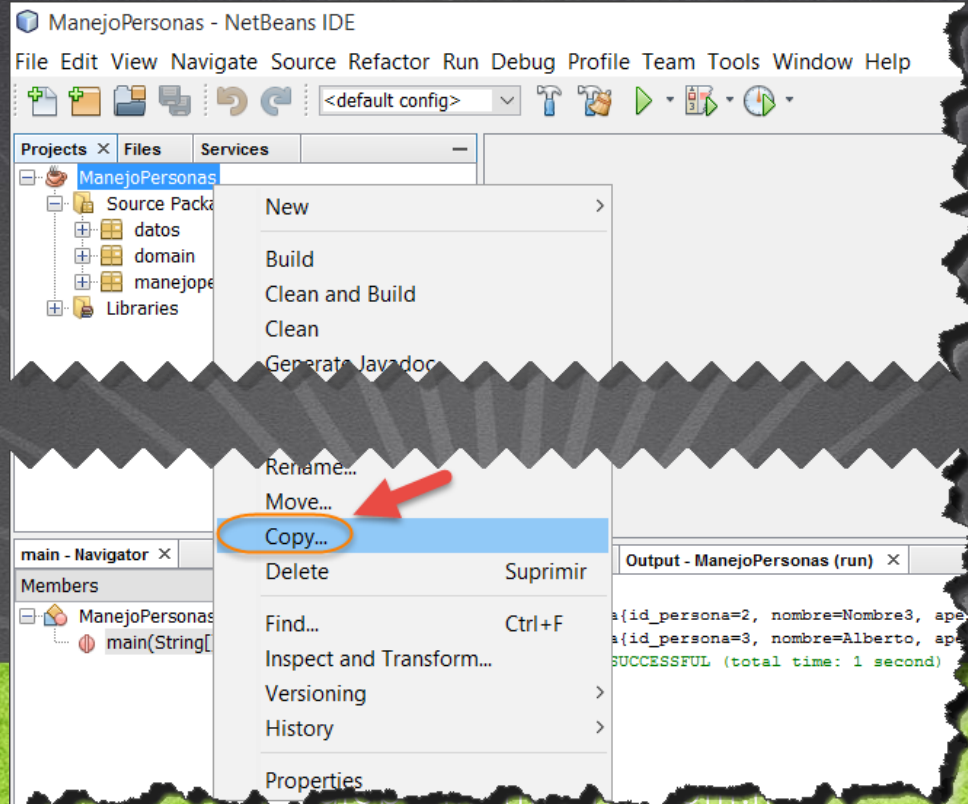
at datos.PersonasJDBC.insert(PersonasJDBC.java:75)

at manejoPersonas.ManejoPersonas.main(ManejoPersonas.java:37)

BUILD SUCCESSFUL (total time: 2 seconds)

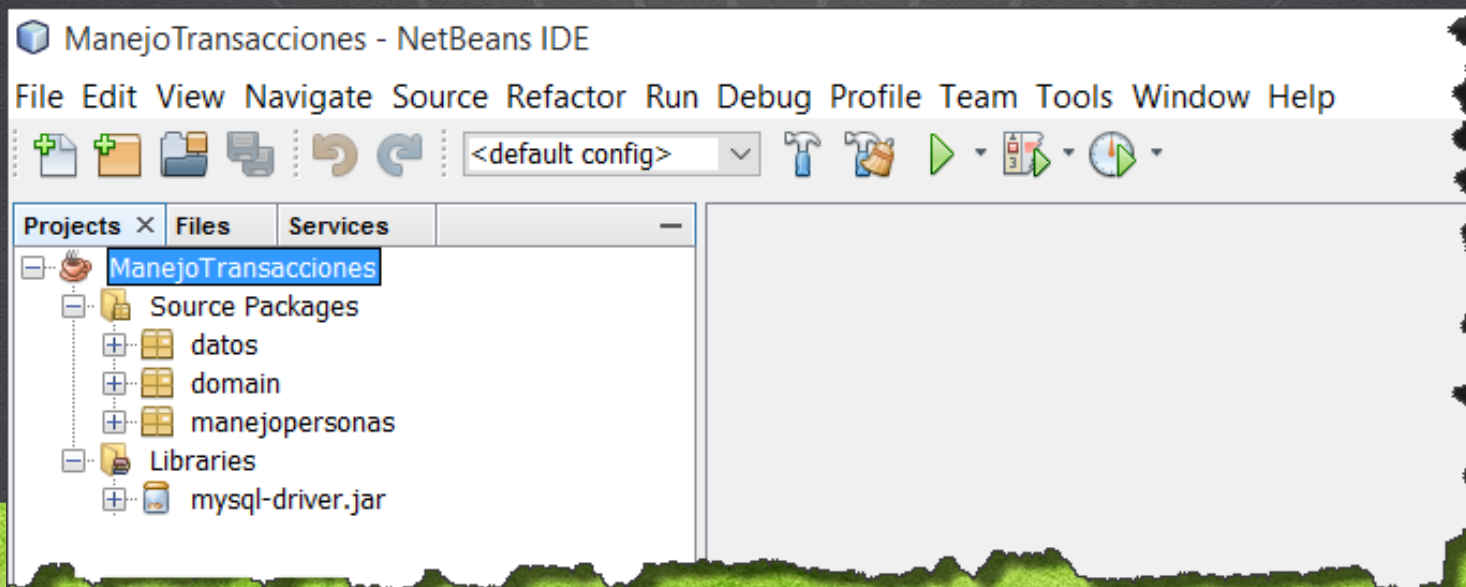
PASO 1. CLONACION DEL PROYECTO

Vamos a copiar el proyecto:



PASO 1. CLONACION DEL PROYECTO (CONT)

El proyecto queda así. Si no es posible copiar el proyecto pueden crear un nuevo proyecto y copiar y pegar los paquetes con el código Java:



PASO 2. MODIFICAMOS EL CÓDIGO

Archivo PersonasJDBC.java:

Dar click para ir al código

```
package datos;

import domain.Persona;
import java.sql.*;
import java.util.*;

public class PersonasJDBC {

    private java.sql.Connection userConn;

    private final String SQL_INSERT =
        "INSERT INTO persona(nombre, apellido) VALUES(?,?)";

    private final String SQL_UPDATE =
        "UPDATE persona SET nombre=?, apellido=? WHERE id_persona=?";

    private final String SQL_DELETE =
        "DELETE FROM persona WHERE id_persona = ?";

    private final String SQL_SELECT =
        "SELECT id_persona, nombre, apellido FROM persona ORDER BY id_persona";
```


PASO 2. MODIFICAMOS EL CÓDIGO (CONT)

Archivo PersonasJDBC.java:

Dar click para ir al código

```
public PersonasJDBC() { }

public PersonasJDBC(Connection conn) {
    this.userConn = conn;
}

public int insert(String nombre, String apellido) throws SQLException {
    Connection conn = null;
    PreparedStatement stmt = null;
    ResultSet rs = null; //no se utiliza en este ejercicio

    int rows = 0; //registros afectados
    try {
        conn = (this.userConn != null) ? this.userConn :
Conexion.getConnection();
        stmt = conn.prepareStatement(SQL_INSERT);
        int index = 1; //contador de columnas
        stmt.setString(index++, nombre); //param 1 => ?
        stmt.setString(index++, apellido); //param 2 => ?
        System.out.println("Ejecutando query:" + SQL_INSERT);
        rows = stmt.executeUpdate(); //no. registros afectados
        System.out.println("Registros afectados:" + rows);

    } finally {
        Conexion.close(stmt);
        if (this.userConn == null) {
            Conexion.close(conn);
        }
    }
    return rows;
}
```

```
public int update(int id_persona, String nombre, String apellido)
throws SQLException {
    Connection conn = null;
    PreparedStatement stmt = null;
    int rows = 0;
    try {
        conn = (this.userConn != null) ? this.userConn :
Conexion.getConnection();
        System.out.println("Ejecutando query:" + SQL_UPDATE);
        stmt = conn.prepareStatement(SQL_UPDATE);
        int index = 1;
        stmt.setString(index++, nombre);
        stmt.setString(index++, apellido);
        stmt.setInt(index, id_persona);
        rows = stmt.executeUpdate();
        System.out.println("Registros actualizados:" + rows);
    } finally {
        Conexion.close(stmt);
        if (this.userConn == null) {
            Conexion.close(conn);
        }
    }
    return rows;
}
```

A CON JDBC

PASO 2. MODIFICAMOS EL CÓDIGO (CONT)

Archivo PersonasJDBC.java:

Dar click para ir al código

```
public int delete(int id_persona) throws SQLException
{
    Connection conn = null;
    PreparedStatement stmt = null;
    int rows = 0;
    try {
        conn = (this.userConn != null) ?
this.userConn : Conexion.getConnection();
        System.out.println("Ejecutando query:" +
SQL_DELETE);
        stmt = conn.prepareStatement(SQL_DELETE);
        stmt.setInt(1, id_persona);
        rows = stmt.executeUpdate();
        System.out.println("Registros eliminados:"
+ rows);
    } finally {
        Conexion.close(stmt);
        if (this.userConn == null) {
            Conexion.close(conn);
        }
    }

    return rows;
}
```

```
public List<Persona> select() throws SQLException {
    Connection conn = null;
    PreparedStatement stmt = null;
    ResultSet rs = null;
    Persona persona = null;
    List<Persona> personas = new ArrayList<>();
    try {
        conn = (this.userConn != null) ? this.userConn : Conexion.getConnection();
        stmt = conn.prepareStatement(SQL_SELECT);
        rs = stmt.executeQuery();
        while (rs.next()) {
            int id_persona = rs.getInt(1);
            String nombre = rs.getString(2);
            String apellido = rs.getString(3);
            persona = new Persona();
            persona.setId_persona(id_persona);
            persona.setNombre(nombre);
            persona.setApellido(apellido);
            personas.add(persona);
        }
    } finally {
        Conexion.close(rs);
        Conexion.close(stmt);
        if (this.userConn == null) {
            Conexion.close(conn);
        }
    }

    return personas;
}
```

PASO 3. MODIFICAMOS EL CÓDIGO

Archivo ManejoPersonas.java:

Dar click para ir al código

```
package manejopersonas;

import datos.Conexion;
import datos.PersonasJDBC;
import java.sql.*;

public class ManejoPersonas {

    public static void main(String[] args) {
        PersonasJDBC personasJDBC = new PersonasJDBC();

        //Creamos un objeto conexion, se va a compartir
        //para todos los queries que ejecutemos
        Connection conn = null;

        try {
            conn = Conexion.getConnection();
            //Revisamos si la conexion esta en modo autocommit
            //por default es autocommit == true
            if (conn.getAutoCommit()) {
                conn.setAutoCommit(false);
            }

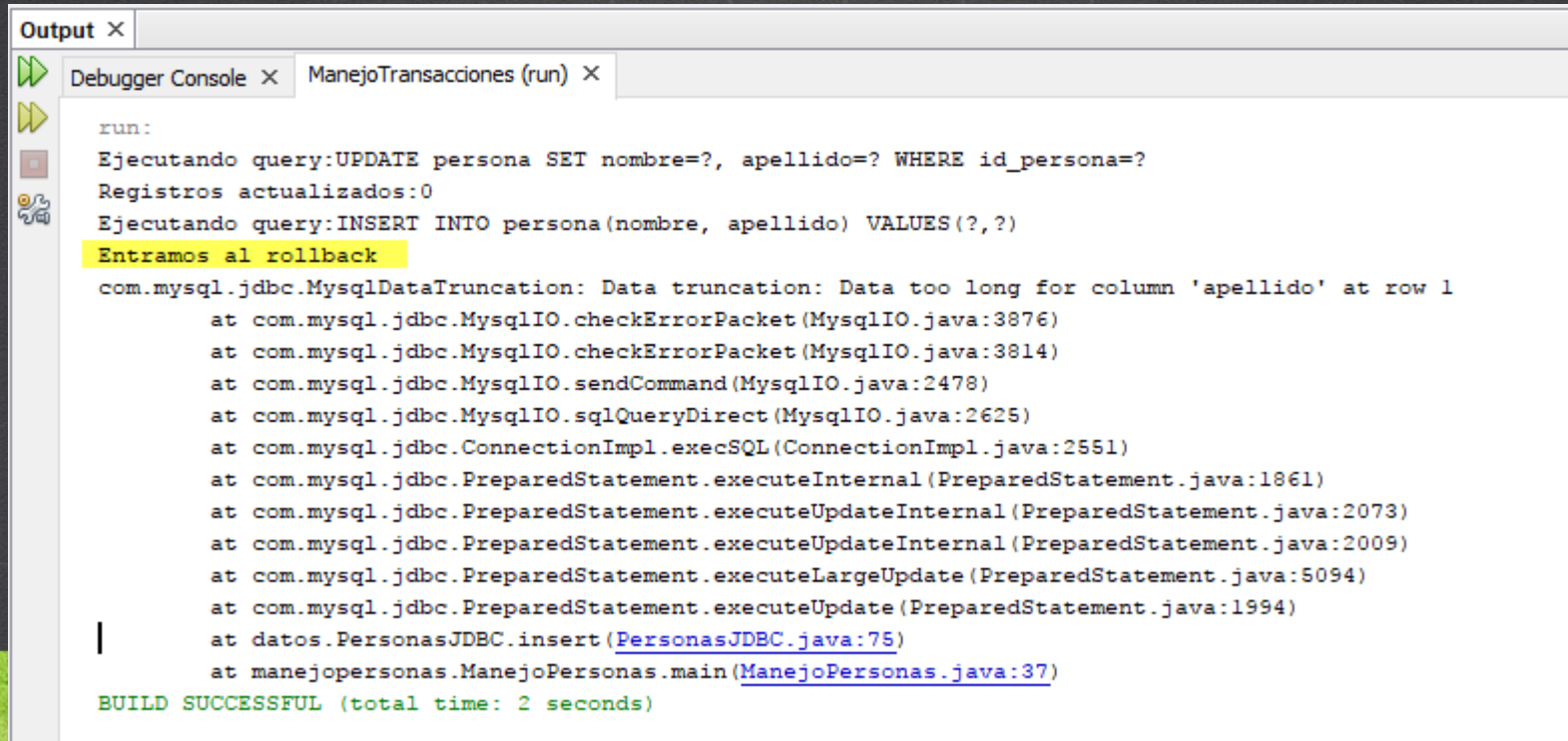
            //Creamos el objeto PersonasJDBC
            //proporcionamos la conexion creada
            PersonasJDBC personas = new PersonasJDBC(conn);
```


Archivo ManejoPersonas.java:

[illegible]

PASO 4. EJECUTAMOS EL PROYECTO (CONT)

El resultado es como sigue:



```
run:
Ejecutando query:UPDATE persona SET nombre=?, apellido=? WHERE id_persona=?
Registros actualizados:0
Ejecutando query:INSERT INTO persona(nombre, apellido) VALUES(?,?)
Entramos al rollback
com.mysql.jdbc.MysqlDataTruncation: Data truncation: Data too long for column 'apellido' at row 1
    at com.mysql.jdbc.MysqlIO.checkErrorPacket(MysqlIO.java:3876)
    at com.mysql.jdbc.MysqlIO.checkErrorPacket(MysqlIO.java:3814)
    at com.mysql.jdbc.MysqlIO.sendCommand(MysqlIO.java:2478)
    at com.mysql.jdbc.MysqlIO.sqlQueryDirect(MysqlIO.java:2625)
    at com.mysql.jdbc.ConnectionImpl.execSQL(ConnectionImpl.java:2551)
    at com.mysql.jdbc.PreparedStatement.executeInternal(PreparedStatement.java:1861)
    at com.mysql.jdbc.PreparedStatement.executeUpdateInternal(PreparedStatement.java:2073)
    at com.mysql.jdbc.PreparedStatement.executeUpdateInternal(PreparedStatement.java:2009)
    at com.mysql.jdbc.PreparedStatement.executeLargeUpdate(PreparedStatement.java:5094)
    at com.mysql.jdbc.PreparedStatement.executeUpdate(PreparedStatement.java:1994)
    at datos.PersonasJDBC.insert(PersonasJDBC.java:75)
    at manejopersonas.ManejoPersonas.main(ManejoPersonas.java:37)
BUILD SUCCESSFUL (total time: 2 seconds)
```

TAREAS EXTRA DEL EJERCICIO

- Probar con distintos valores y verificar el resultado.



Experiencia y Conocimiento para tu vida

CURSO DE JAVA CON JDBC

www.globalmentoring.com.mx

CONCLUSIÓN DEL EJERCICIO

- Con este ejercicio hemos puesto en práctica el concepto de transacciones en JDBC.
- Hemos visto que al ejecutar una sentencia que provoca un error, podemos hacer un rollback de toda la transacción y por lo tanto no afectaremos a el estado de la base de datos.
- Para más información de este tema:
 - <https://docs.oracle.com/javase/tutorial/jdbc/basics/transactions.html>

CURSO ONLINE

JAVA CON JDBC

Por: Ing. Ubaldo Acosta



Experiencia y Conocimiento para tu vida

CURSO DE JAVA CON JDBC

www.globalmentoring.com.mx